

200

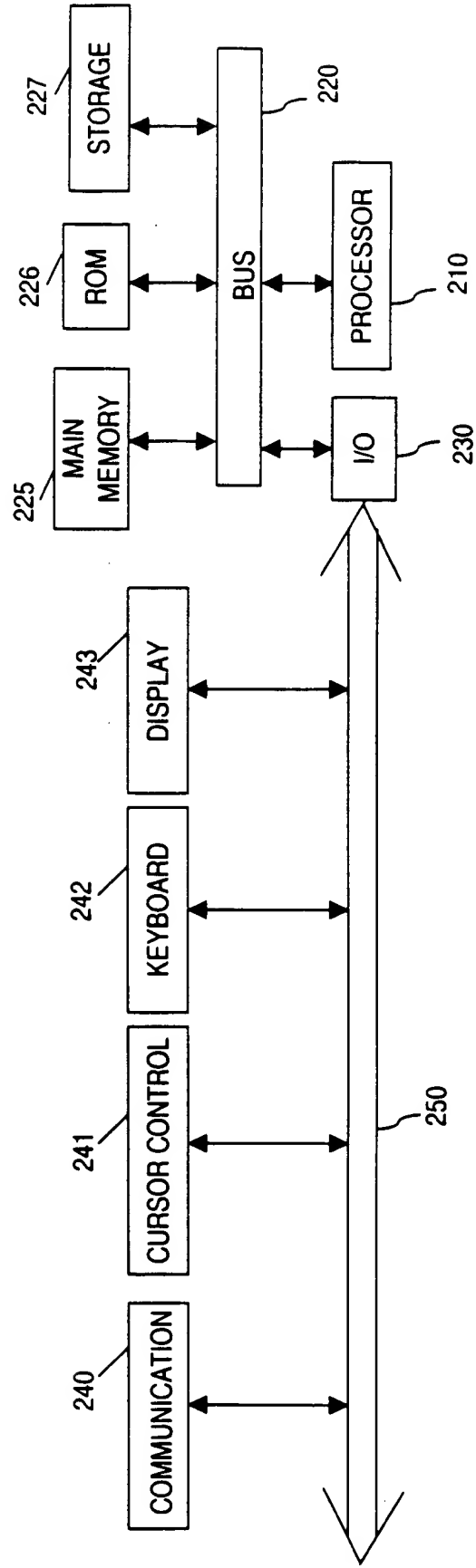


FIG. 1

A diamond-shaped network diagram with nodes labeled PURE, R, O, I, RO, RI, OI, ROI, and LOST. Each node is associated with a numerical value and underlines. The connections are as follows: PURE is connected to R, O, and I; R is connected to RO and O; O is connected to RO, RI, and I; I is connected to OI; RO is connected to ROI; RI is connected to ROI; OI is connected to ROI; and ROI is connected to LOST. An arrow points to the ROI node.

Node	Value
PURE	~ <u>295</u>
R	<u>280</u> ~
O	~ <u>285</u>
I	~ <u>290</u>
RO	~ <u>275</u>
RI	~ <u>265</u>
OI	~ <u>270</u>
ROI	~ <u>260</u>
LOST	~ <u>255</u>

FIG. 2

250

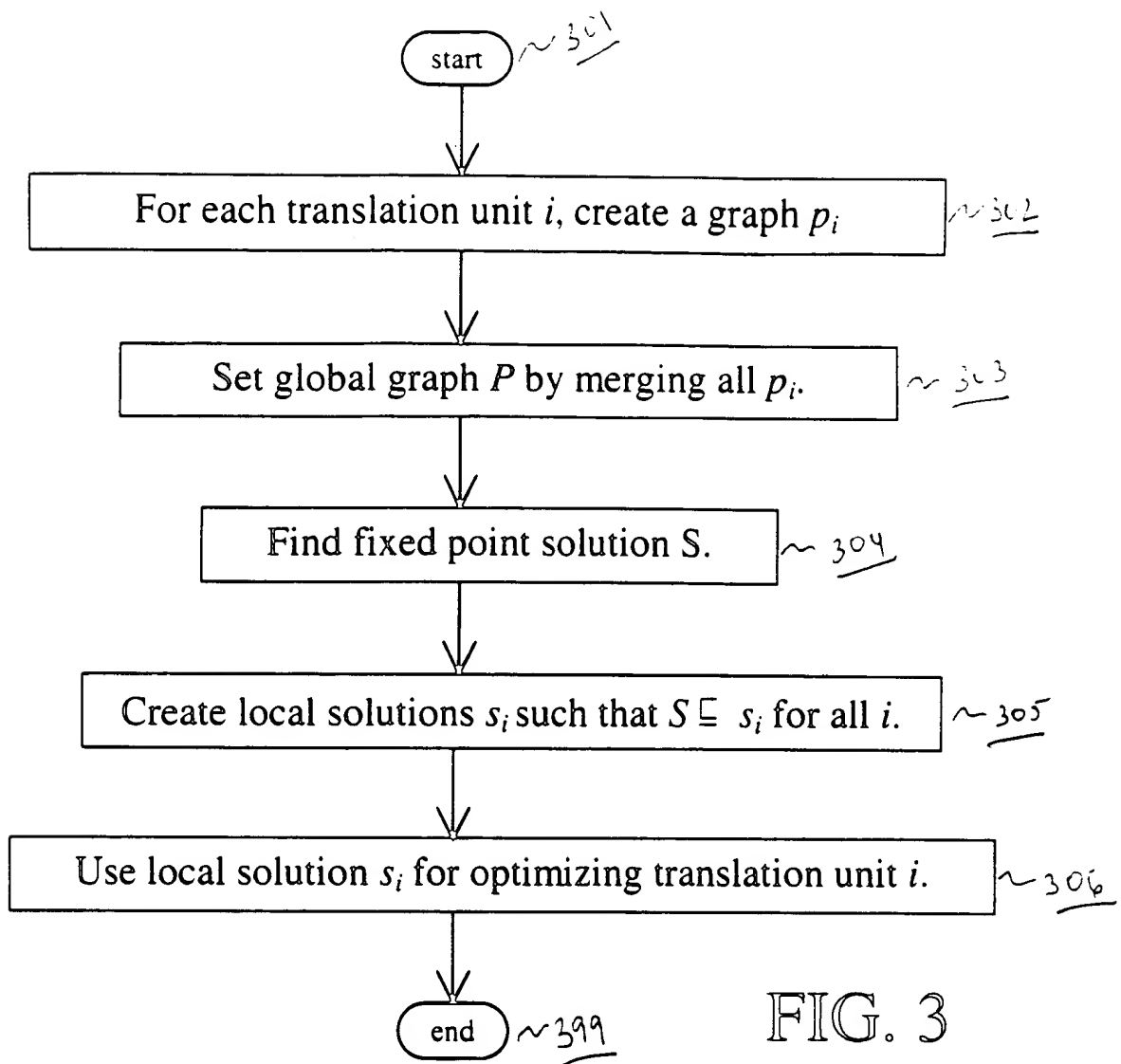
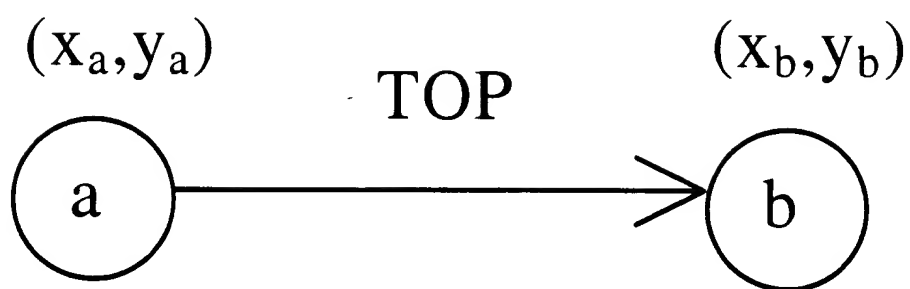


FIG. 3

# FIG. 4A

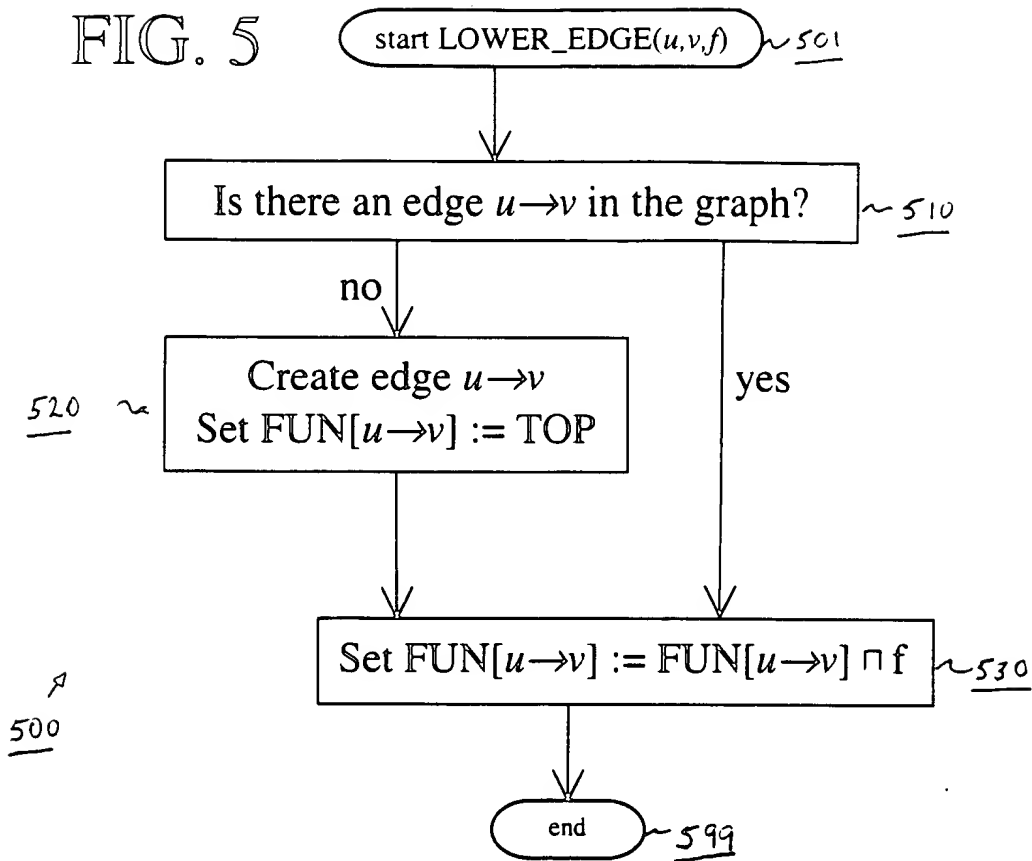
Function	Function (x,y)
<u>410</u> ~ TOP	(PURE,PURE)
<u>420</u> ~ COPY	(y,y)
<u>430</u> ~ IN_TO_LOST	if $y \leq I \Rightarrow (LOST,LOST)$ otherwise $\Rightarrow (PURE,PURE)$
<u>440</u> ~ UNRETURN	if $y=LOST \Rightarrow (LOST,LOST)$ otherwise $\Rightarrow (z,z)$ where $z=y \sqcup OI$
<u>450</u> ~ COPY_AND_IN_TO_LOST	if $y \leq I \Rightarrow (LOST,LOST)$ otherwise $\Rightarrow (y,y)$
<u>460</u> ~ CAT_FORMAL	(y,PURE)
<u>470</u> ~ CAT_ACTUAL	(PURE,y)
<u>480</u> ~ GATE	if $x=LOST \Rightarrow (LOST,LOST)$ else if $x \leq R \Rightarrow (z,z)$ where $z = (x \sqcup OI) \sqcap y$ else $(z,z)$ where $z = (x \sqcup OI)$

FIG. 4A



**FIG. 4B**

FIG. 5



600 ↗

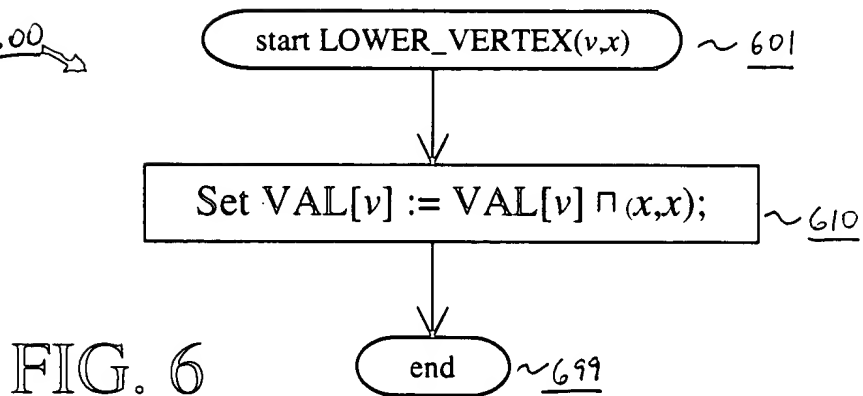


FIG. 6

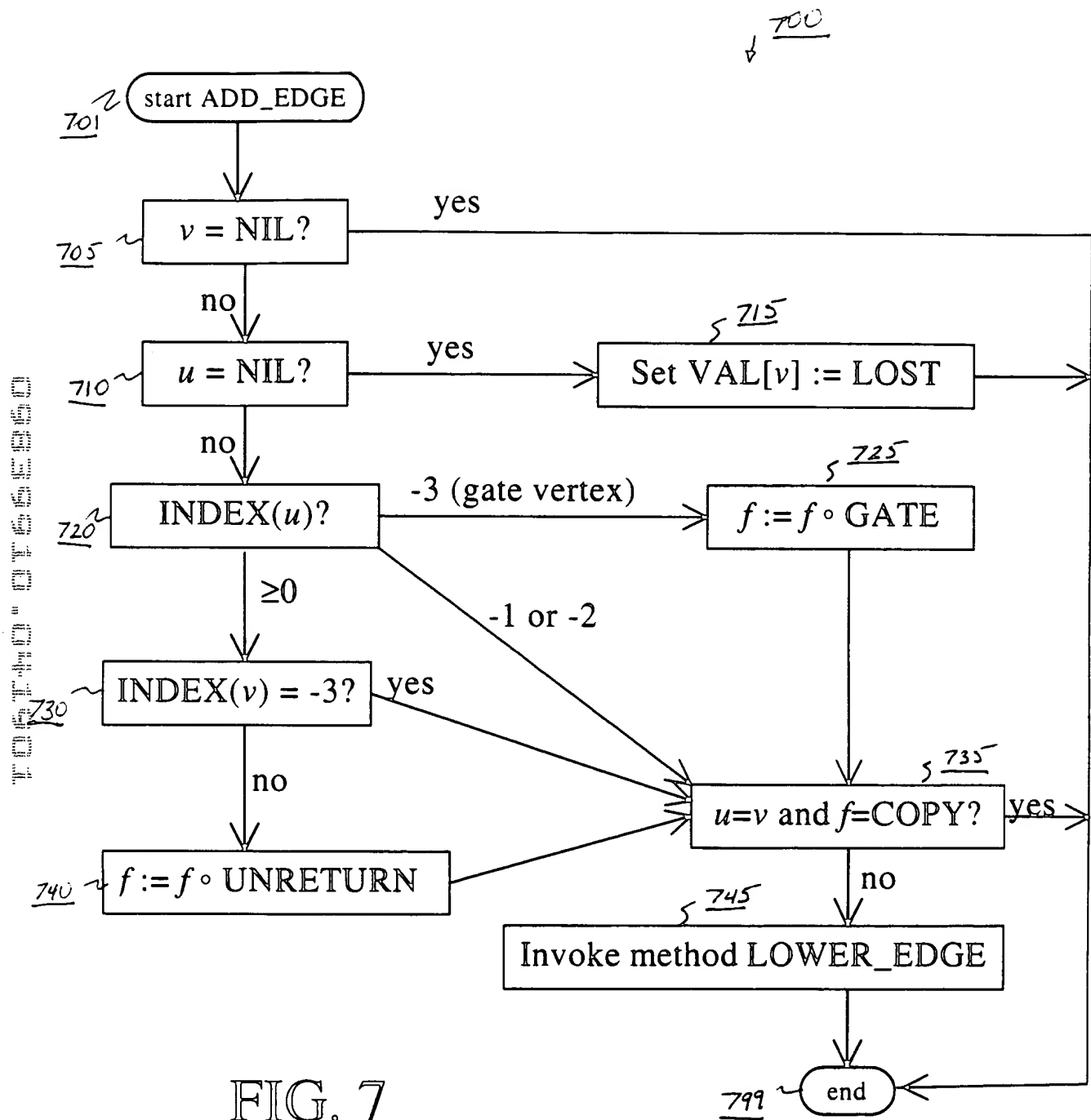
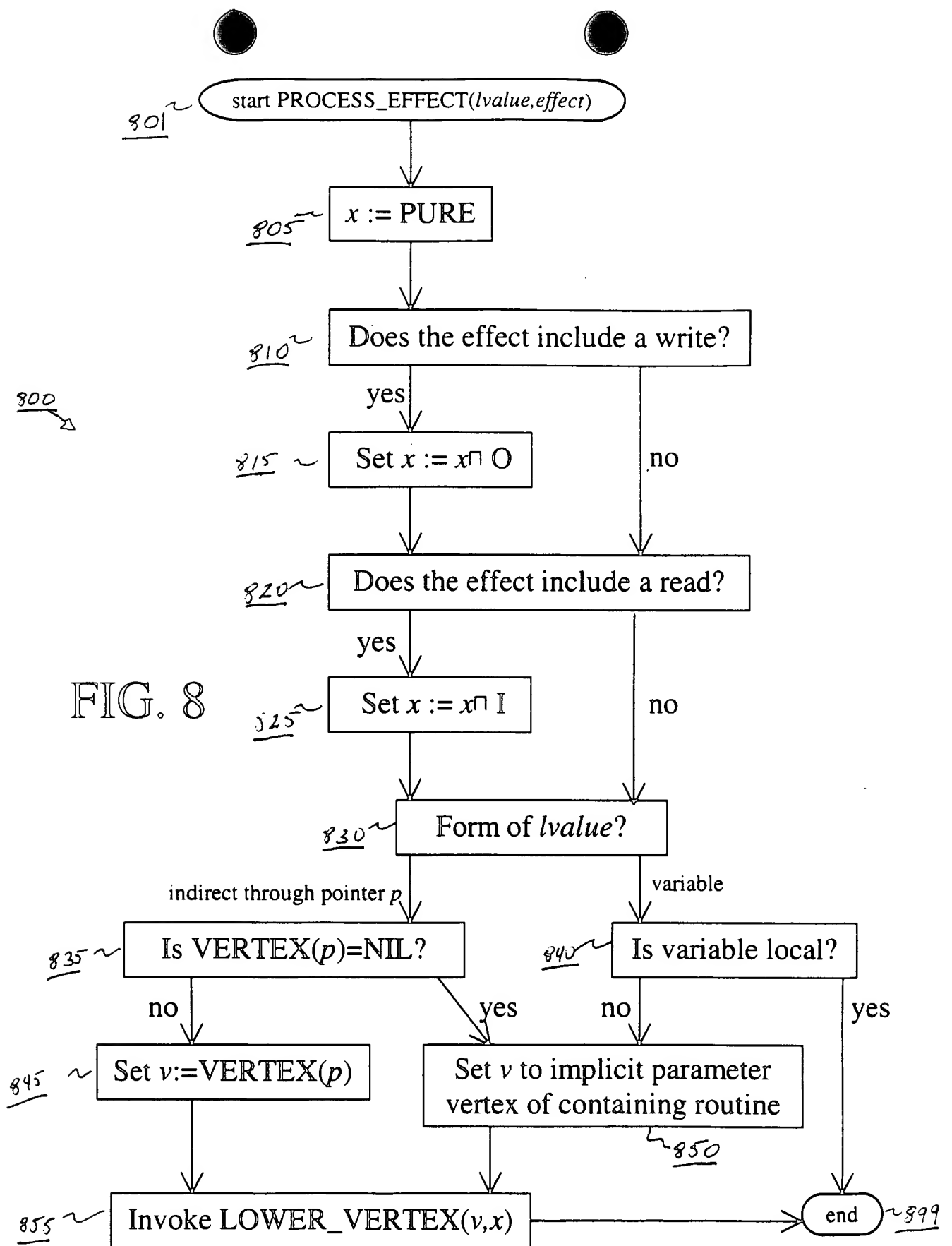
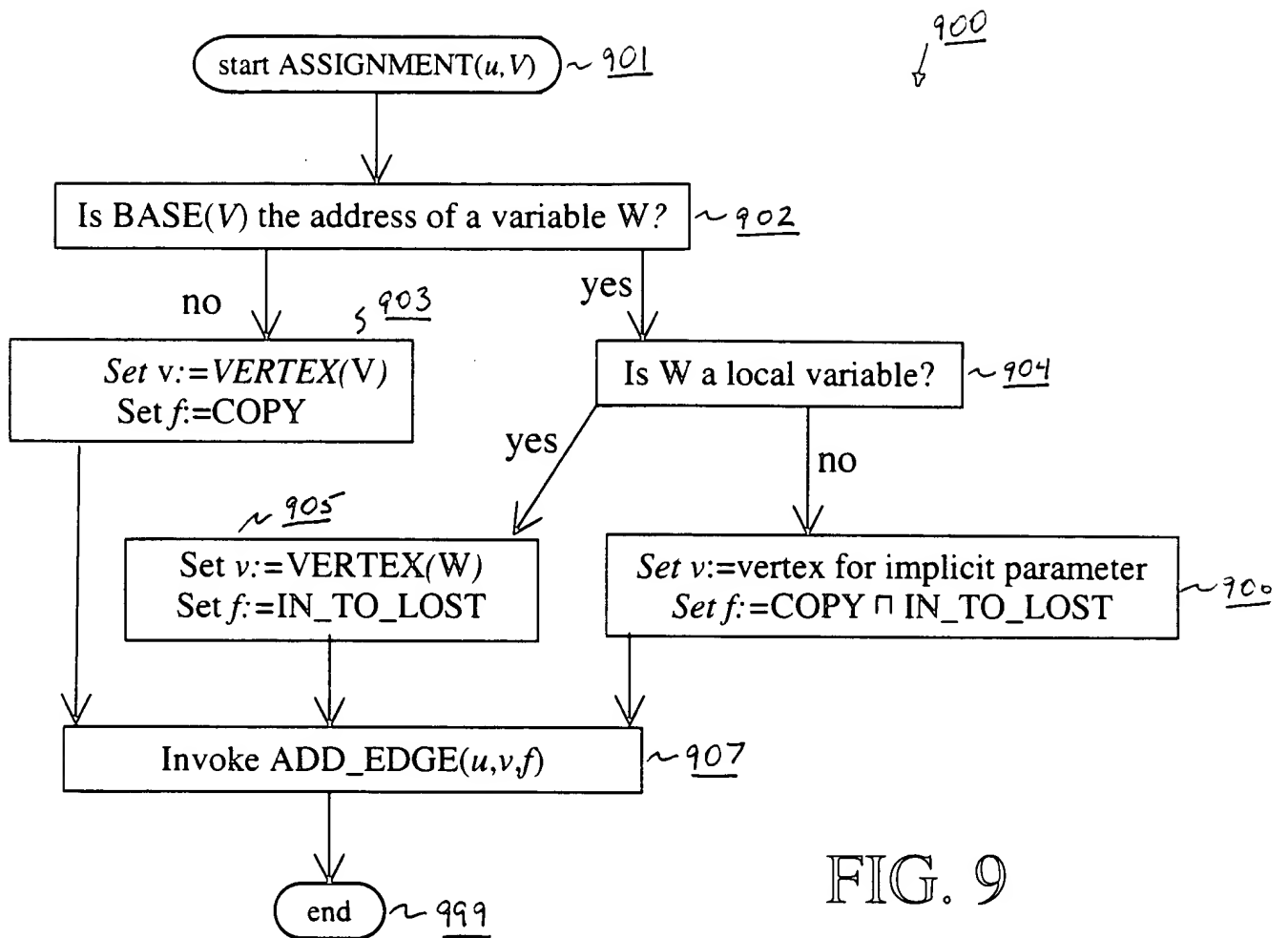


FIG. 7







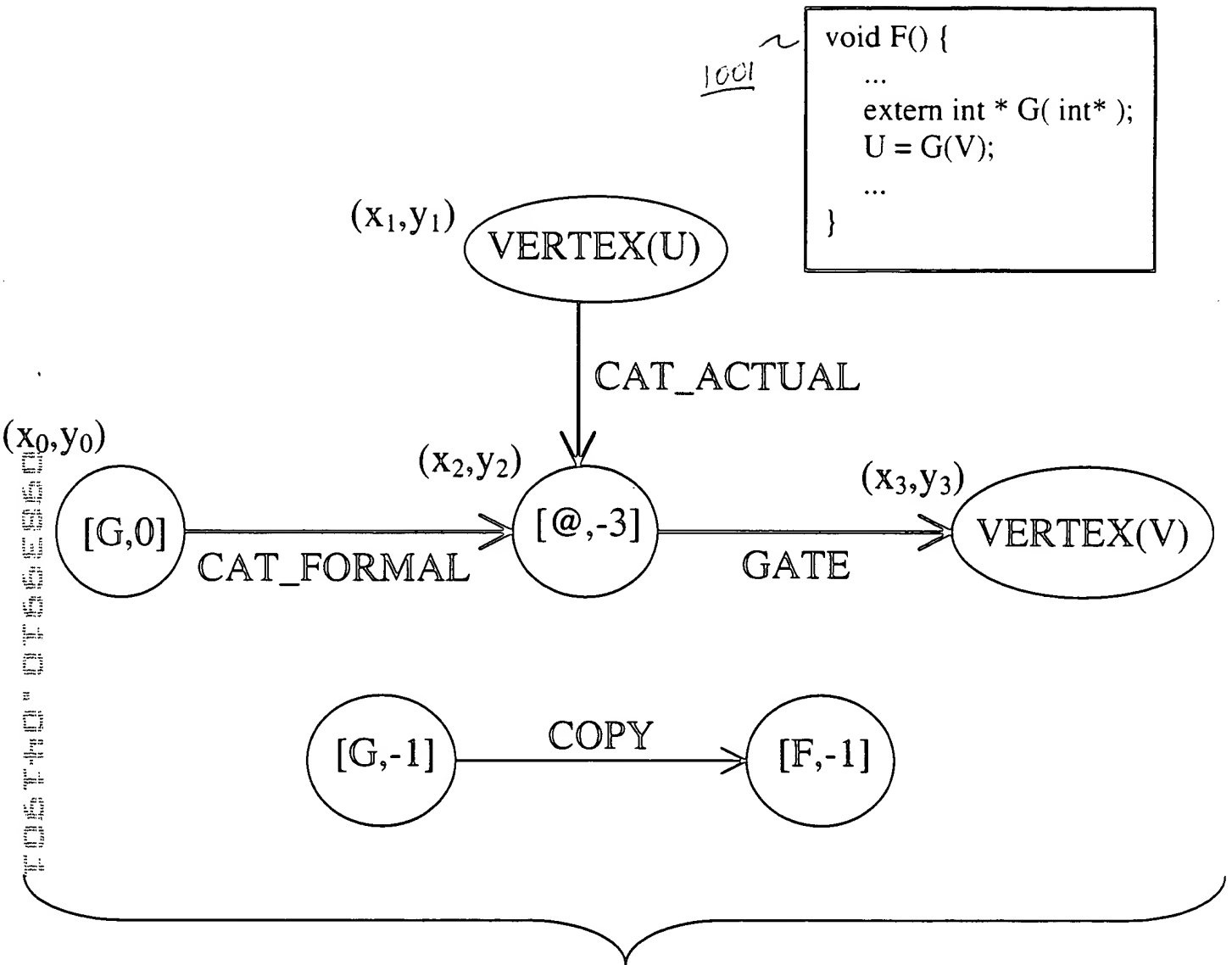


FIG. 10

// Translation unit #1

```
int* f( int* a, int* b, int n ) {  
    int *c = a;  
    if( n>0 ) {  
        int* d = a+1;  
        int* e = b+1;  
        int* z = f( d, e, n-1 );  
        c = z-1;  
        *c = *b;  
    }  
    return c;  
}
```

// Translation unit #2

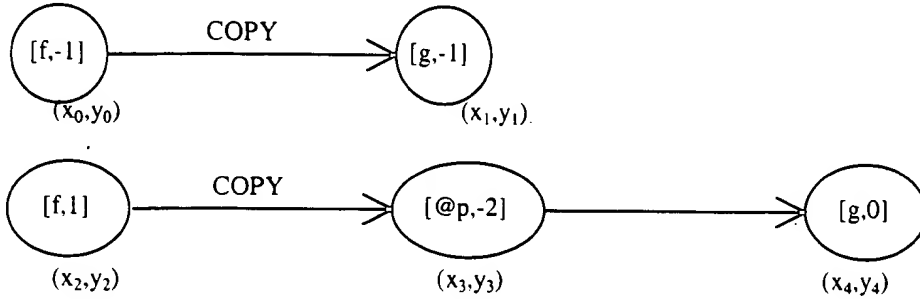
extern int\* f(int\* a, int\* b, int n );

```
void g( int* p ) {  
    int y[10];  
    f( &y[0], p, 10 );  
}
```

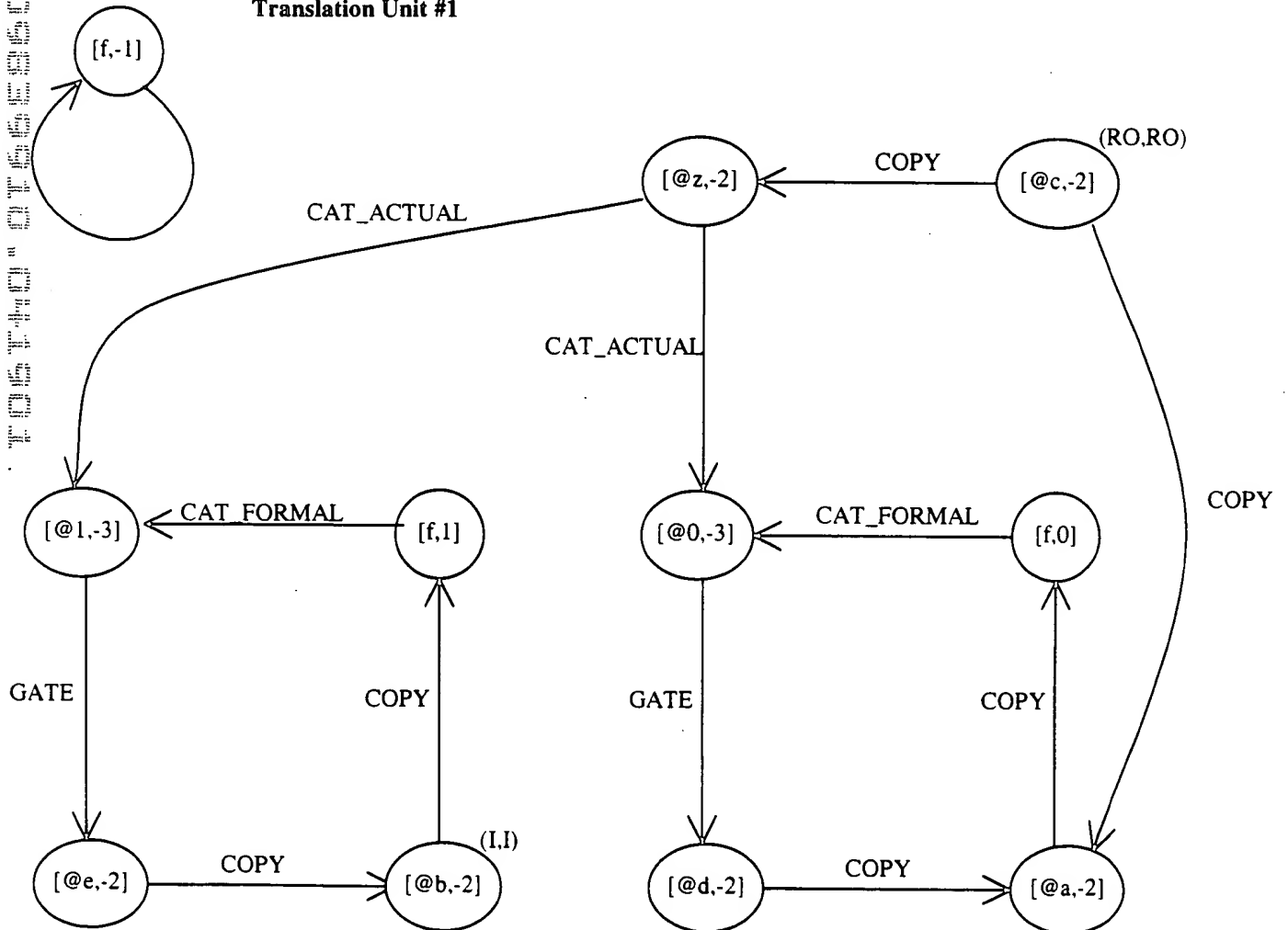
FIG. 11

# FIG. 12

## Translation Unit #2



## Translation Unit #1



Translation Unit #1	
Source line	Action
(entry into f)	Add [@a,-2] → [f,0] Add [@b,-2] → [f,1]
int *c = a;	Add [@c,-2] → [@a,-2]
n>0	None
int *d = a+1;	Add [@d,-2] → [@a,-2]
int *e = b+1;	Add [@e,-2] → [@b,-2]
int* z = f(d,e,n-1)	Add [@z,-2] → [@0,-3] → [@d,-2] Add [f,0] → [@0,3] Add [@z,-2] → [@1,-3] → [@b,-2] Add [f,1] → [@1,3] Add [f,-1] → [f,-1]
c = z-1;	Add [@c,-2] → [@z,-2]
*c = *b;	Lower VAL[@c,-2] to O Lower VAL[@b,-2] to I
return c;	Lower VAL[@c,-2] to R

Translation Unit #2	
Action	Action
int *p = &x[0];	None
for( int i=0; i<10; i++)	None (no pointer assignments)
*p = i;	Lower VAL[@p,-2] to O
p=p+1	None (edge omitted by self-loop rule)
c = z-1;	Add [@c,-2] → [@z,-2]

FIG. 13